Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра компьютерных систем в управлении и проектировании

(КСУП)

Разработка плагина «Ладья» для САПР «КОМПАС-3D» v.20

Пояснительная записка по дисциплине

«Основы разработки САПР» (ОРСАПР)

Выполнил:

Студент гр.589-2

\_\_\_\_\_\_\_Радченко С.Н.

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2022 г.

Руководитель:

к.т.н., доцент

каф. КСУП

\_\_\_\_\_\_\_Калентьев А.А.

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2022 г.

**РЕФЕРАТ**

Лабораторная работа 31 с., 16 рис., 8 источников.

Ключевые слова: КОМПАС-3D, VISUAL STUDIO, C#, КОМПАС API, ПЛАГИН, САПР.

Целью данной работы является разработка плагина для автоматизации проектирования модели шахматной ладьи.

Результатом работы является плагин, осуществляющий построение ладьи по заданным пользователям параметрам.

Пояснительная записка выполнена в текстовом редакторе Microsoft Word 2016.

**Содержание**

[1 Введение 4](#_Toc93431739)

[2 Описание API 5](#_Toc93431740)

[3 Обзор аналогов 11](#_Toc93431741)

[4 Описание предмета проектирования 12](#_Toc93431742)

[4.1 Выбор инструментов и средств реализации 14](#_Toc93431743)

[4.2 Назначение плагина 14](#_Toc93431744)

[5 Описание реализации 15](#_Toc93431745)

[5.1 Диаграмма классов 15](#_Toc93431746)

[6 Описание программы для пользователя 19](#_Toc93431747)

[7 Тестирование плагина 22](#_Toc93431748)

[7.1 Функциональное тестирование 22](#_Toc93431749)

[7.2 Модульное тестирование 24](#_Toc93431750)

[7.3 Нагрузочное тестирование 27](#_Toc93431751)

[Заключение 29](#_Toc93431752)

[Список использованных источников 30](#_Toc93431753)

# Введение

В настоящее время автоматизация – основной способ повышения производительности и эффективности труда инженерно-технических работников, занимающихся моделированием сложных устройств. Под автоматизацией проектирования понимается такой способ выполнения процесса разработки проекта, когда проектные процедуры операции осуществляются проектировщиком при тесном взаимодействии с компьютером. Автоматизация проектирования предполагает систематическое использование средств вычислительной техники при рациональном распределении функций между проектировщиком и ком­пьютером и обоснованном выборе методов машинного решения задач.

Цель автоматизации проектирования - повышение качества, снижение материальных затрат, сокращение сроков проектирования и числа проектировщиков, повышение производительности их труда.

Система автоматизированного проектирования (САПР) – это организационно-техническая система, состоящая из комплекса средств автоматизации проектирования и обеспечивающая автоматизированное проектирование на всех этапах разработки прибора – от проектных исследований, проектирования до подготовки производства [1]

Целью данной работы является разработка плагина, автоматизирующего построение модели «Ладья» для системы автоматизированного проектирования КОМПАС-3D с помощью интегрированной среды разработки Visual Studio 2019 Сommunity. [2]

Интегрированная среда разработки Visual Studio — это стартовая площадка для написания, отладки и сборки кода, а также последующей публикации приложений. Интегрированная среда разработки (IDE) представляет собой многофункциональную программу, которую можно использовать для различных аспектов разработки программного обеспечения.

# Описание API

Сегодня встречаются задачи, решение которых не реализовано в CAD-системах. Чаще всего это очень узкоспециализированные задачи, которые встречаются на каком-то конкретном предприятии или подотрасли. Для решения подобных задач есть возможность использовать КОМПАС-3D как платформу и на базе него создать свое приложение, которое позволит автоматизировать решение таких задач. Для создания таких приложений в КОМПАС-3D есть открытый API[3].

Главным интерфейсом API системы КОМПАС является KompasObject.[4] Методы этого интерфейса, реализуют наиболее общие функции работы с документами системы, системными настройками, файлами, а также дают возможность получить указатели на другие интерфейсы (интерфейсы динамического массива, работы с математическими функциями, библиотек моделей или фрагментов и различных структур параметров определенного типа).

Ниже в таблице 1.1 представлены основные свойства и методы интерфейса KompasObject.

Таблица 1.1 – Методы и свойства интерфейса KompasObject

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Входные параметры** | **Тип возвращаемых данных** | **Описание** |
| Document3D() | – | ksDocument | Метод для получения указателя на интерфейс трехмерного графического документа (детали или сборки) |

Продолжение таблицы 1.1

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Входные параметры** | **Тип возвращаемых данных** | **Описание** |
| GetParamStruct  (short structType) | structType – тип интерфейса параметров | StructType2D | Метод для получения указателя на интерфейс графического документа (чертежа или фрагмента) |
| Visible | – | bool | Свойство видимости приложения |
| Quit() | – | – | Метод для закрытия активного окна приложения |

В таблице 1.2 представлены методы интерфейса ksEntity, необходимые для разработки плагина.

Таблица 1.2 – Методы интерфейса ksEntity

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип возвращаемых данных** | **Описание** |
| Create() | bool | Создать объект в модели |
| GetDefinition() | IUnkown | Получить указатель на интерфейс параметров объектов и элементов |
| Update() | bool | Изменить свойства объекта (используя ранее установленные свойства) |
| NewEntity (short objType) | указатель на интерфейс ksEntity или IEntity. | Реальный объект создается в модели после вызова метода [Create](mk:@MSITStore:D:\KOMPAS\SDK\SDK.chm::/ksEntity_Create.htm). |

В таблице 1.3 представлены свойства и методы интерфейса ksDocument2D, необходимые для разработки плагина.

Таблица 1.3 – Методы интерфейса ksDocument2D

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Входные параметры** | **Тип возвращаемых данных** | **Описание** |
| ksRectangle(  ksRectangle-Param param, int style) | param – параметры прямоугольника.  style – стиль линии. | int | Получить указатель на прямоугольник на двумерной плоскости либо 0 в случае ошибки |
| ksCircle(  double xc, double yc, double rad, int style) | xc, yc – координаты центра окружности.  rad – радиус окружности.  style – стиль линии. | int | Получить указатель на окружность на двумерной плоскости либо 0 в случае ошибки |

В таблице 1.4 представлены свойства и методы интерфейса ksDocument3D, необходимые для разработки плагина.

Таблица 1.4 – Методы интерфейса ksDocument3D

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Входные параметры** | **Тип возвращаемых данных** | **Описание** |
| Create (bool invisible, bool \_typeDoc) | invisible – признак режима редактирования документа (true – невидимый режим, false –  видимый режим), typeDoc – тип документа (true – деталь, false – сборка). | bool | Создать документ-модель (деталь или сборку) |
|  |  |
| GetPart(int type) | type – тип компонента из перечисления Типы компонентов. | ksPart | Получить указатель на интерфейс компонента в соответствии с заданным типом |

В таблице 1.5 представлены методы интерфейса ksPart, необходимые для разработки плагина.

Таблица 1.5 – Свойства и методы интерфейса ksPart.

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Входные параметры** | **Тип возвращаемых данных** | **Описание** |
| EntityCollection  (short objType) | objType – тип объектов, содержащихся в массиве. | ksEninty-  Collection | Формирует массив объектов и возвращает указатель на его интерфейс |
| GetDefaultEntity( short objType) | objType – тип объекта | ksEntity | Получить указатель на интерфейс объекта, создаваемого системой по умолчанию |
| GetPart(int type) | Type – тип компонента | ksPart | Получить указатель на интерфейс компонента в соответствии с заданным типом |
| NewEntity(short objType) | objType – тип объекта | ksEntity | Создать новый интерфейс объекта и получить указатель на него |

В таблице 1.6 представлены типы объектов документа-модели, необходимые для разработки плагина.

Таблица 1.6 – Некоторые типы объектов документа-модели

|  |  |  |
| --- | --- | --- |
| **Идентификатор объекта** | **Название объекта** | **Интерфейс параметров** |
| o3d\_unknown | Неизвестный (включает все объекты) |  |
| o3d\_planeXOZ | Плоскость XOZ | ksPlaneParam |
| o3d\_planeYOZ | Плоскость YOZ | ksPlaneParam |
| o3d\_planeXOY | Плоскость XOY | ksPlaneParam |
| o3d\_sketch | Эскиз | ksSketchDefinition |
| o3d\_face | Грань | ksFaceDefinition |
| o3d\_cutExtrusion | Вырезать выдавливанием | ksCutExtrusionDefinition |
| o3d\_fillet | Операция "скругление" | ksFilletDefinition |

# Обзор аналогов

MechaniCS — это приложение к САПР AutoCAD и Inventor, предназначенное для разработки и оформления чертежей и спецификаций в соответствии с ЕСКД, проектирования изделий общего машиностроения.

Приложение MechaniCS располагает стандартным набором инструментов для оформления чертежей по ЕСКД. В нем удобно создавать форматы и штампы чертежей, проставлять размеры и шероховатость поверхностей, допуски размеров, формы и расположения, делать выноски и различные специальные обозначения. При помощи маркеров пользователи имеют возможность создавать собственные элементы оформления чертежей. Есть инструменты создания спецификаций. Специальные средства разработаны для проектирования типовых механических соединений. Кроме того, в приложение включены методики расчета различных механических характеристик деталей машин. База элементов содержит стандартные и унифицированные врезаемые элементы, детали и сборочные единицы, которые можно использовать при проектировании.

CSoft MechaniCS обеспечивает специалиста всем необходимым для проектирования машиностроительных объектов: более чем двумя тысячами стандартов (включая ГОСТ, ОСТ, DIN и ISO) и унифицированными компонентами, возможностью создавать собственные интеллектуальные объекты, выполнять инженерные расчеты с отображением результатов на модели, оформлять проекции чертежей по ЕСКД и многим другим. Все детали общей конструкторско-технологической базы обладают интеллектом и являются объектно-зависимыми. При изменении параметров одной детали все связанные с ней объектно-зависимые детали изменятся автоматически, причем в соответствии с их параметрами в базе. Такая технология — мощный инструмент многовариантного проектирования, залог повышения качества выпускаемых проектов. Важно, что этот подход одинаково доступен пользователям Autodesk AutoCAD и Autodesk Inventor [3].

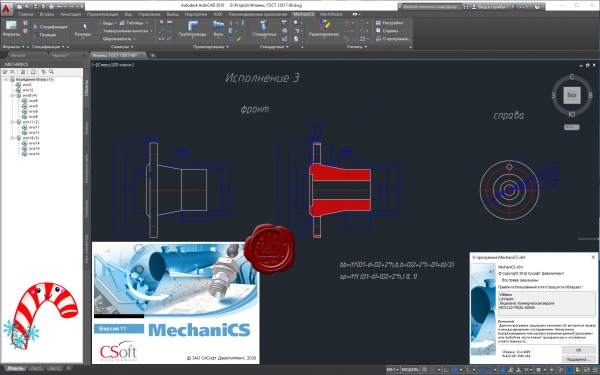


Рисунок 1.1 – Скриншот приложения MechaniCS

# Описание предмета проектирования

Предметом проектирования является шахматная ладья.

По желанию клиента в создаваемой ладье могут изменяться параметры, приведенные ниже:

1. A — высота фигуры (70 – 600 мм);
2. B — Диаметр нижнего основания (20 – 200 мм);
3. C — Диаметр верхнего основания (15 – 100 мм);
4. D — Высота нижнего основания (15 – 100 мм);
5. E — Высота верхнего основания (14 – 90 мм).

Плагин имеет следующие зависимости:

1. Диаметр нижнего основания (B) должно быть больше диаметра верхнего основания (C);

2. Высота нижнего основания (D) должно быть больше высоты верхнего основания (E);

3. Следующая сумма: (D + 2\*E) не может быть больше общей высоты фигуры (A).

На рисунке 4.1 показан общий вид шахматной ладьи:

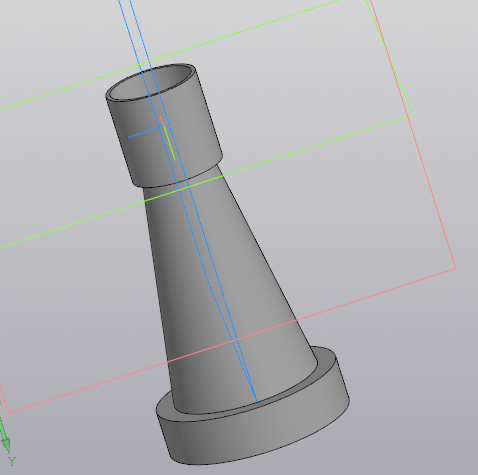


Рисунок 4.1 – Общий вид ладьи

На рисунках 4.2 – 4.4 представлены различные виды ладьи с указанными параметрами:

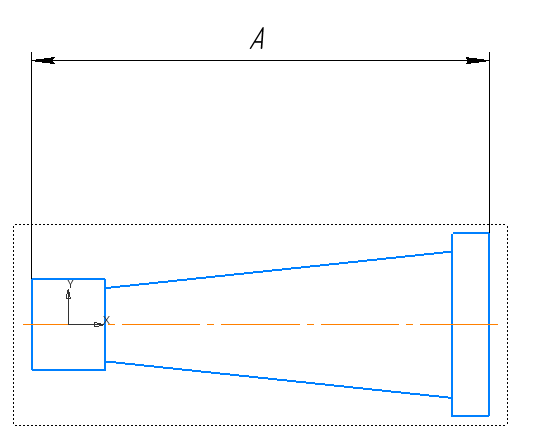


Рисунок 4.2 – Вид сбоку

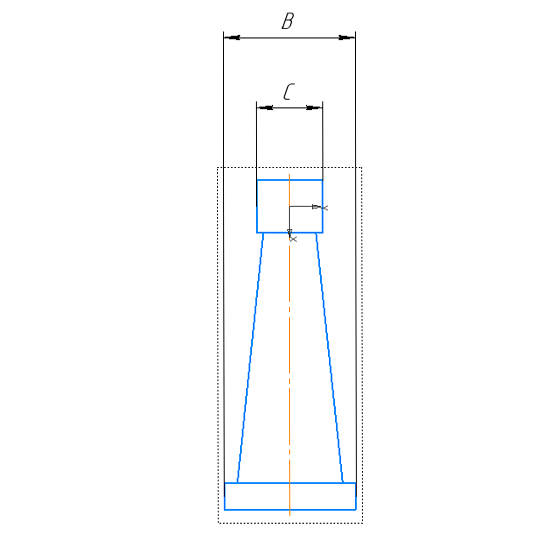


Рисунок 4.3 – Вид сбоку

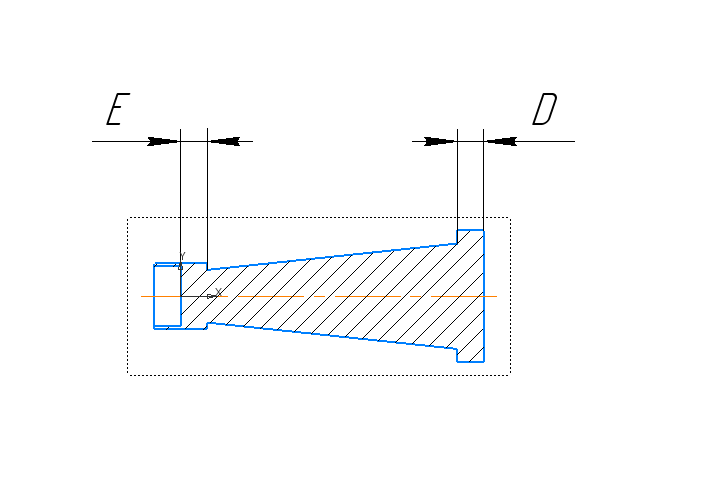


Рисунок 4.3 – Вид сбоку в разрезе

## Выбор инструментов и средств реализации

На основе требований к техническому заданию программа выполнена на языке программирования C# в среде Microsoft Visual Studio 2019 с использованием .NET Framework 5.0 [2], библиотека «Kompas6API5» [5] для основных операций в САПР КОМПАС-3D.

Инструментом тестирования и создания модульных тестов был выбран тестовый фреймворк NUnit [6] версии 3.13.2.

Технология разработки графического интерфейса: Windows Forms [7].

## Назначение плагина

Назначение разрабатываемого плагина обусловлено быстрым моделированием шахматной ладьи. Благодаря данному расширению, проектировщики могут наглядно рассмотреть спроектированную модель, при необходимости перестроить под необходимые им параметры.

# Описание реализации

Для графического описания абстрактной модели проекта, а также пользовательского взаимодействия (сценарии действия) использован стандарт UML.

UML язык графического описания для объектного моделирования в области разработки программного обеспечения. UML является языком широкого профиля, это – открытый стандарт, использующий графические обозначения для создания абстрактной модели системы, называемой UML – моделью. UML был создан для определения, визуализации, проектирования и документирования, в основном, программных систем. UML не является языком программирования, но на основании UML возможна генерация кода и наоборот.

При использовании UML были простроена диаграмма классов.

## Диаграмма классов

Диаграмма классов – структурная диаграмма языка моделирования UML, демонстрирующая общую структуру иерархии классов системы, их коопераций, атрибутов (полей), методов, интерфейсов и взаимосвязей между ними.

На рисунке 5.1 представлена изначальная диаграмма классов.

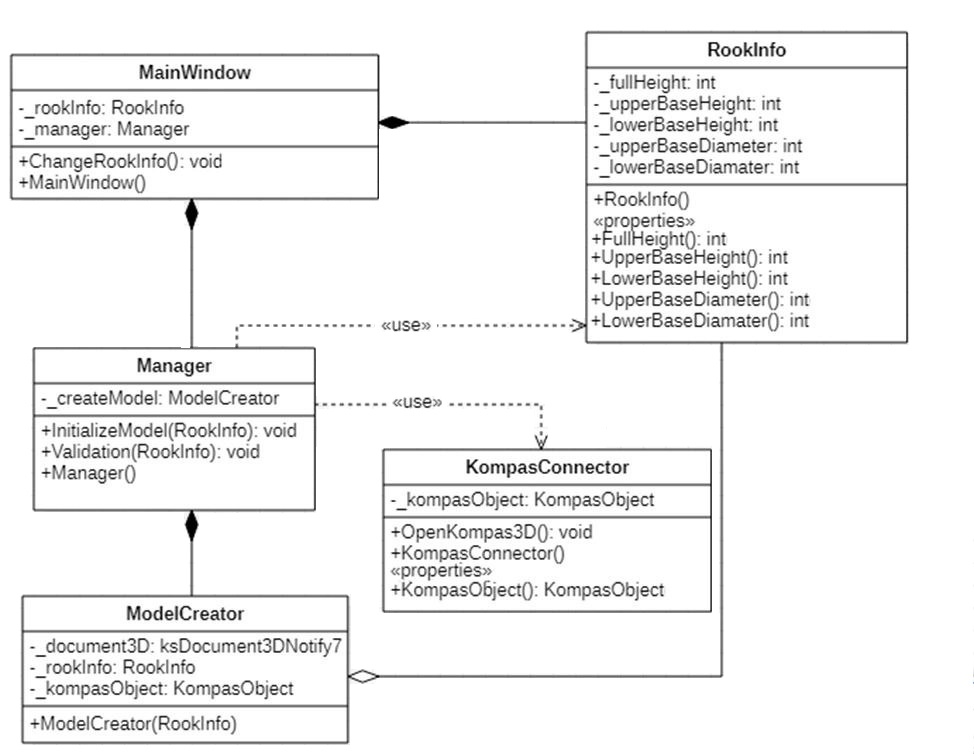


Рисунок 5.1 – Изначальная диаграмма классов

В таблице 5.1 представлено описание полей и методов класса MainWindow.

Таблица 5.1 – Описание полей и методов класса MainWindow.

|  |  |
| --- | --- |
| **Название** | **Описание** |
| \_rookInfo: RookInfo | Данные для построения ладьи |
| \_manager: Manager | Взаимодействие между плагинов и КОМПАС 3D |
| MainWindow() | Конструктор |
| ChangeRookInfo() | Изменение данных для построения ладьи |

В таблице 5.2 представлено описание свойств и методов класса RookInfo.

Таблица 5.2 – Описание свойств и методов класса RookInfo.

|  |  |
| --- | --- |
| **Название** | **Описание** |
| FullHeight: int | Полная высота фигуры |
| UpperBaseHeight: int | Высота верхнего основания |
| LowerBaseHeight: int | Высота нижнего основания |
| UpperBaseDiameter: int | Диаметр верхнего основания |
| LowerBaseDiameter: int | Диаметр нижнего основания |
| RookInfo() | Конструктор |

В таблице 5.3 представлено описание полей и методов класса Manager.

Таблица 5.3 – Описание полей и методов класса Manager.

|  |  |
| --- | --- |
| **Название** | **Описание** |
| \_createModel: CreatingModel | Создание модели |
| InitializeModel(RookInfo):void | Инициализация модели |
| Manager() | Конструктор |
| Validator(InfoRook): void | Валидация введенных данных |

В таблице 5.4 представлено описание полей и методов класса CreatingModel.

Таблица 5.4 – Описание полей и методов класса CreatingModel.

|  |  |
| --- | --- |
| **Название** | **Описание** |
| \_document3D:ksDocument3DNotify7 | Документ, содержащий 3D-модель или сборку |
| \_rookInfo: RookInfo | Данные для построения ладьи |
| CreatingModel(RookInfo) | Конструктор |

В таблице 5.5 представлено описание полей и методов класса KompasConnector.

Таблица 5.5 – Описание свойств и методов класса KompasConnector.

|  |  |
| --- | --- |
| **Название** | **Описание** |
| openKompas3D(): void | Открытие Компаса |
| KompasConnector() | Конструктор |
| GetKompasObject() | Получение KompasObject |
| \_kompasObject: KompasObject | Интерфейс API-системы КОМПАС |

На рисунке 5.2 представлена конечная uml-диаграмма классов системы.

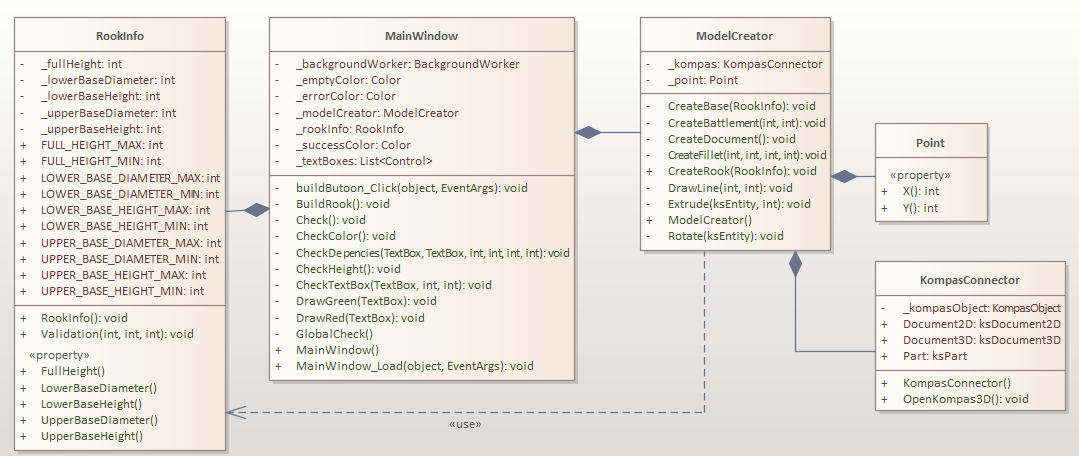


Рисунок 5.2 – Итоговая диаграмма классов

В процессе разработки была переосмыслена изначальная диаграмма классов, в результате чего:

* Решено было не реализовывать класс Manager, так как при разработке системы оказалось, что в он будет содержать только объект класса ModelCreator, что является совершенно излишним;
* В класс RookInfo были добавлены константы с диапазонами для задания параметров ладьи;
* Был добавлен вспомогательный класс Point для упрощения создания 2D-эскиза;
* В классе ModelCreator были реализованы методы, отвечающие за различные операции, которые используются при построении ладьи для того, чтоб не загромождать один метод InitializeModel() (из первоначальной схемы);
* Так же некоторые названия методов/параметров были изменены для большего соответствия настоящему назначению этих компонентов.

# Описание программы для пользователя

Пользовательский интерфейс состоит из окна, в котором вводятся данные для построения шахматной ладьи. Если все данные были введены корректно, то кнопка «Построить» будет разблокирована и пользователь может построить модель.

Так же под списком параметров расположены переключатели выбора дополнительной функциональности плагина – можно сделать основание ладьи квадратным или добавить прорези на башне.

На рисунке 6.1 представлен макет интерфейса программы.

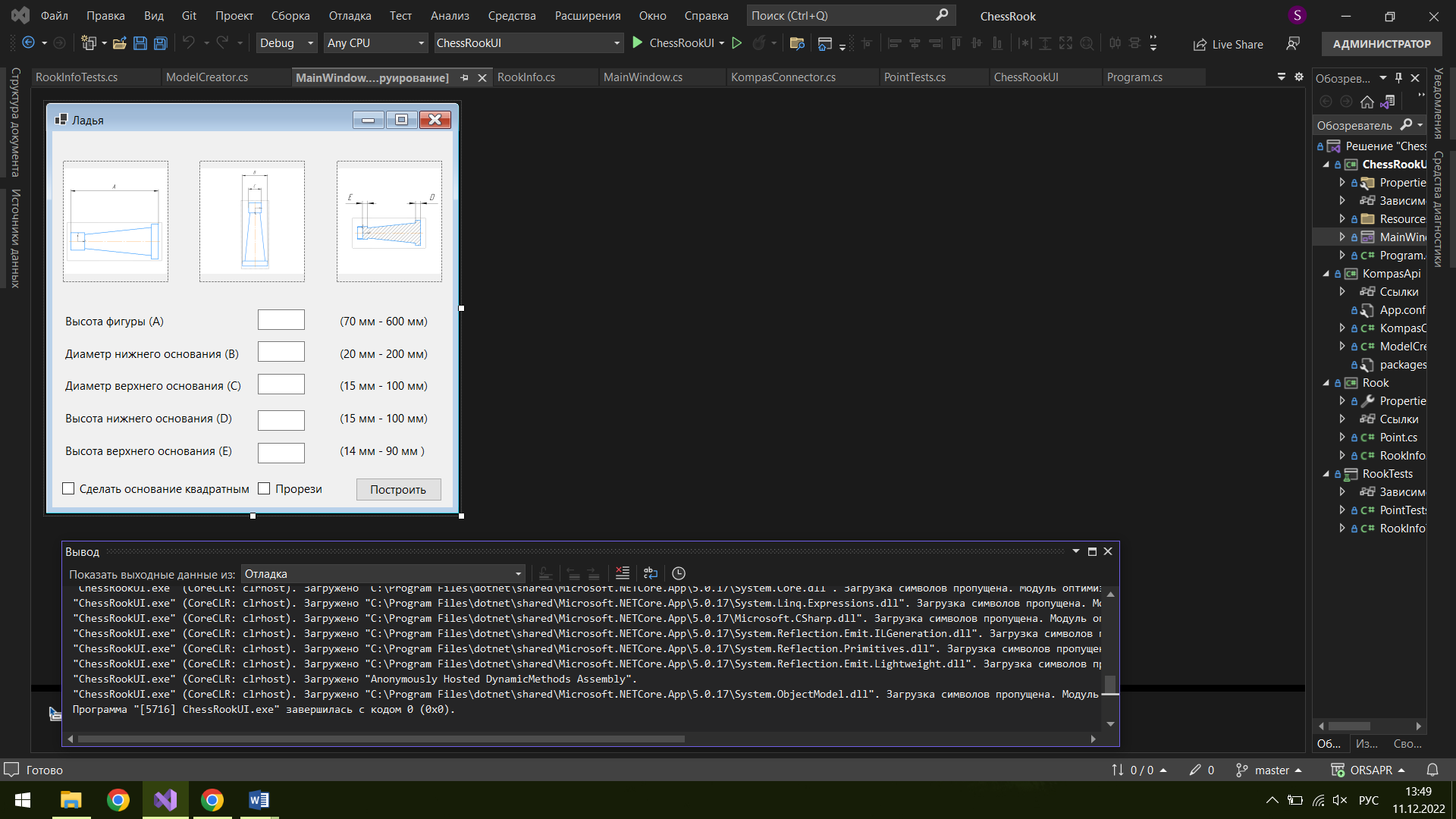


Рисунок 6.1 – Пользовательский интерфейс

Если ввести неверные параметры, то кнопка «Построить» будет заблокирована для нажатия, а поля с неправильными данными будут выделены красным цветом. Так же добавлены подсказки, которые появляются при наведении на соответствующее текстовое поле. На рисунке 6.2 представлен макет интерфейса с некорректно введенными данными.

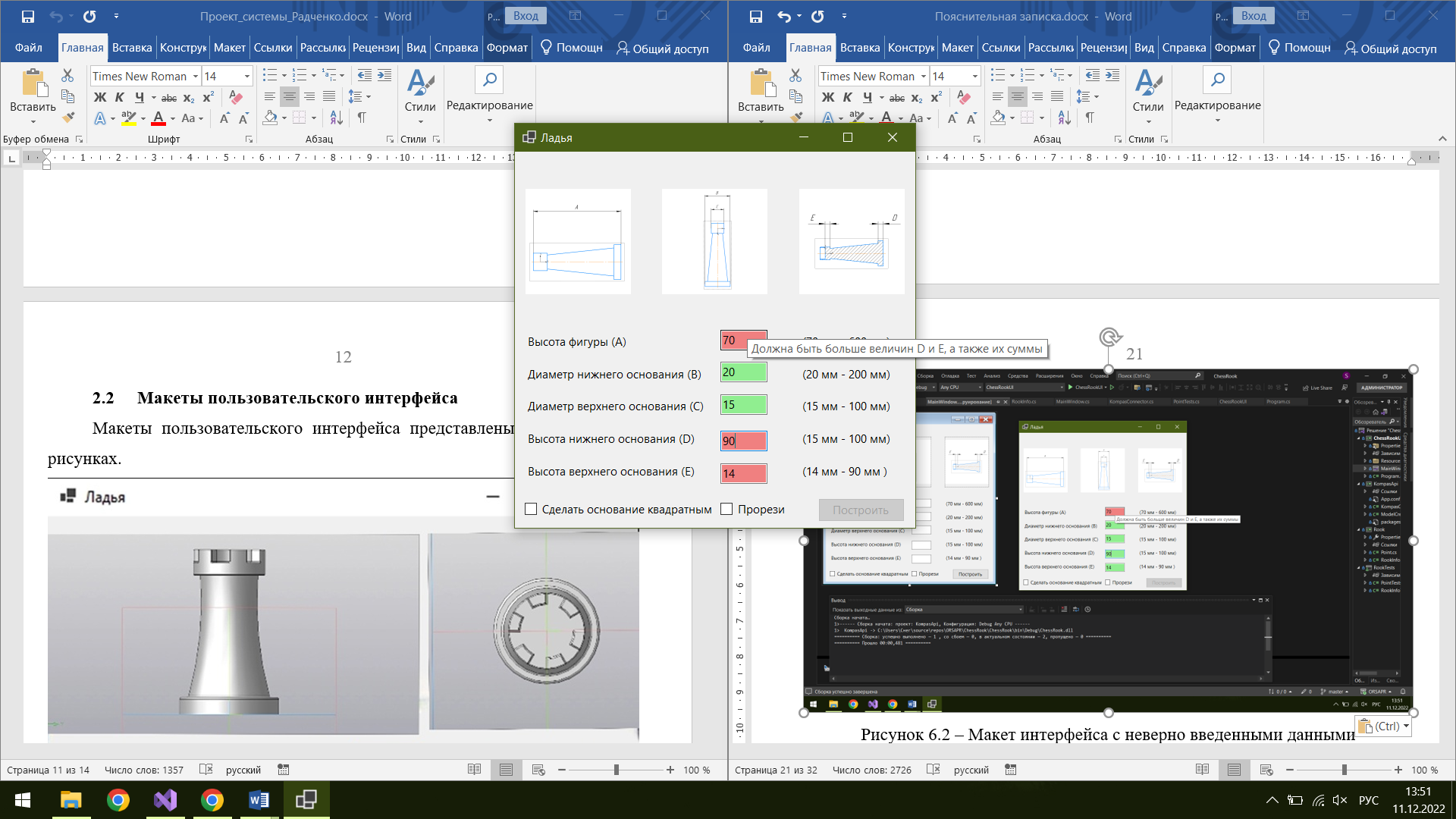


Рисунок 6.2 – Макет интерфейса с неверно введенными данными

После ввода корректных параметров кнопка «Построить» будет снова доступна и при ее нажатии будет построена модель ладьи.

Ладья, построенная по заданным параметрам по умолчанию без дополнительных функциональностей представлена на рисунке 6.3

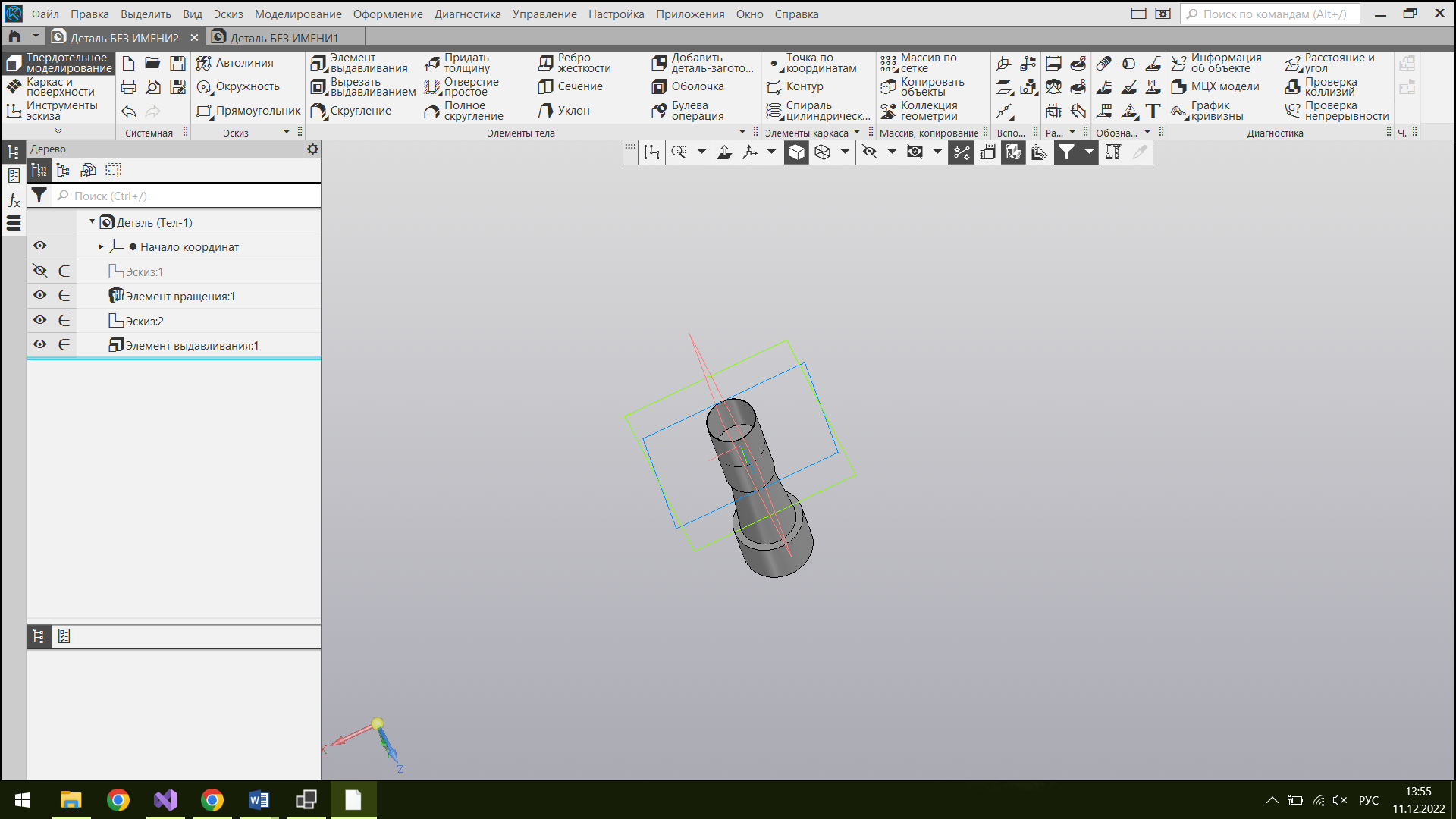


Рисунок 6.3 – Модель ладьи, построенная по параметрам по умолчанию

Модель ладьи с квадратным основанием и прорезями представлена на рисунке 6.4.

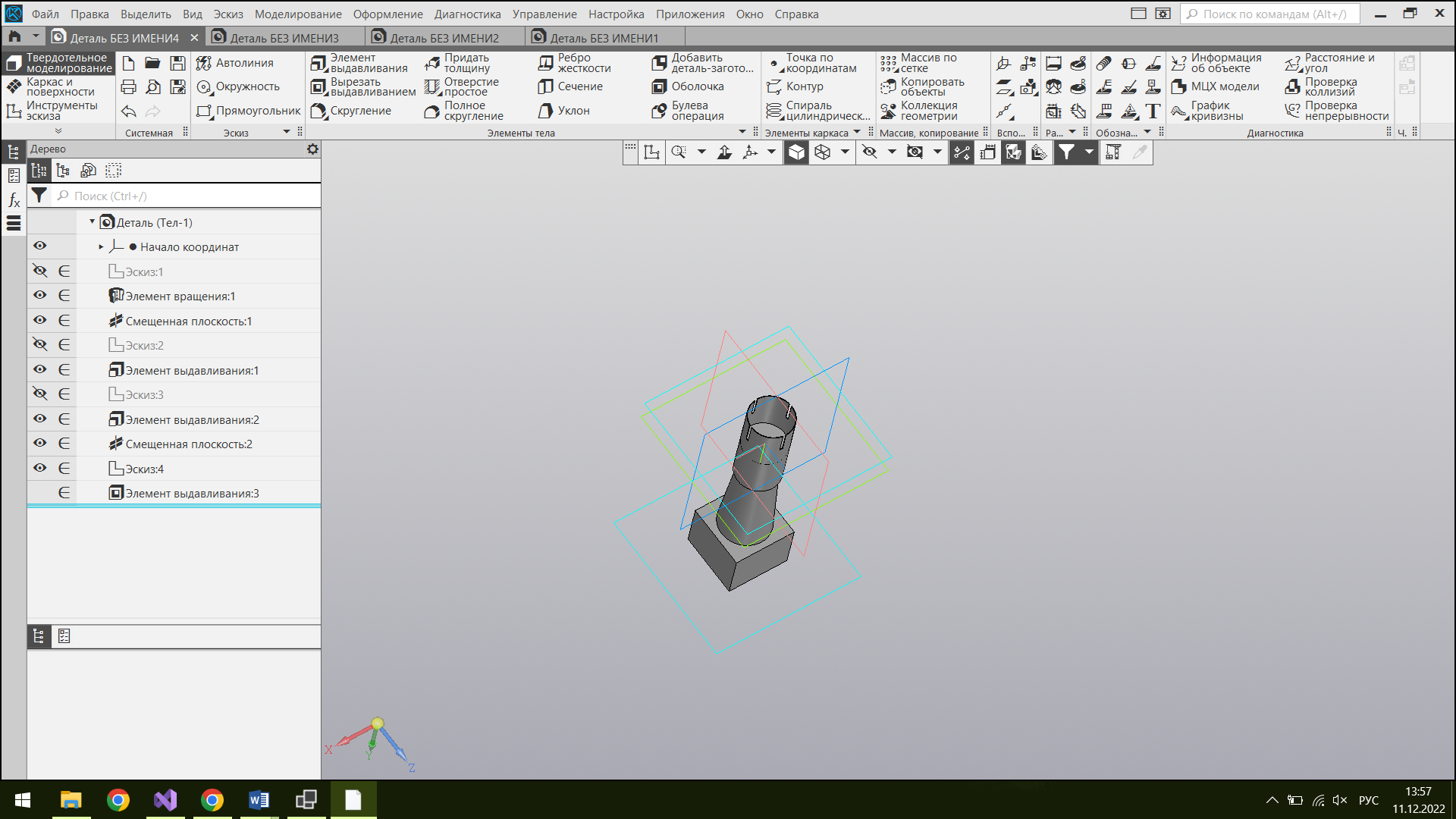


Рисунок 6.4 – Модель ладьи с квадратным основанием и прорезями

Модель ладьи с круглым основанием и прорезями представлена на рисунке 6.5.

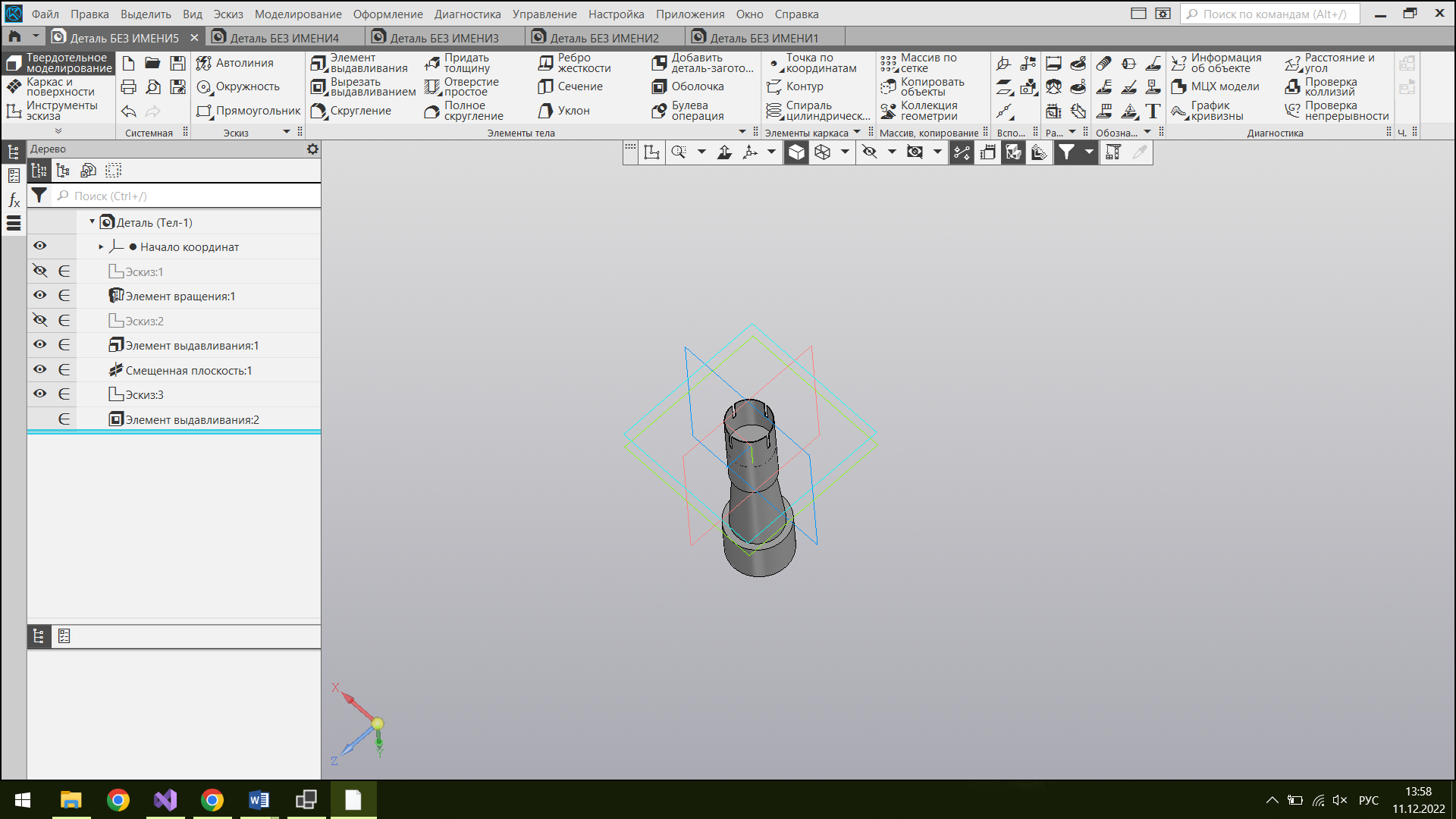


Рисунок 6.5 – Модель ладьи с круглым основанием и прорезями

# Тестирование плагина

Тестирование позволяет убедиться в работоспособности программы, выявлять ошибки при изменении какого-либо функционала.

## Функциональное тестирование

При функциональном тестировании проверялось корректность работы плагина, а именно, соответствие полученного результата в виде трехмерной модели, с входными параметрами.

Проведено тестирование максимальных и минимальных параметров модели.

На рисунке 7.1 представлена модель с минимально введенными параметрами.

Минимальные параметры модели:

– Высота фигуры (А) – 70 мм;

– Диаметр нижнего основания (B) – 20 мм;

– Диаметр верхнего основания (C) – 15 мм;

– Высота нижнего основания (D) – 15 мм;

– Высота верхнего основания (E) – 14 мм.

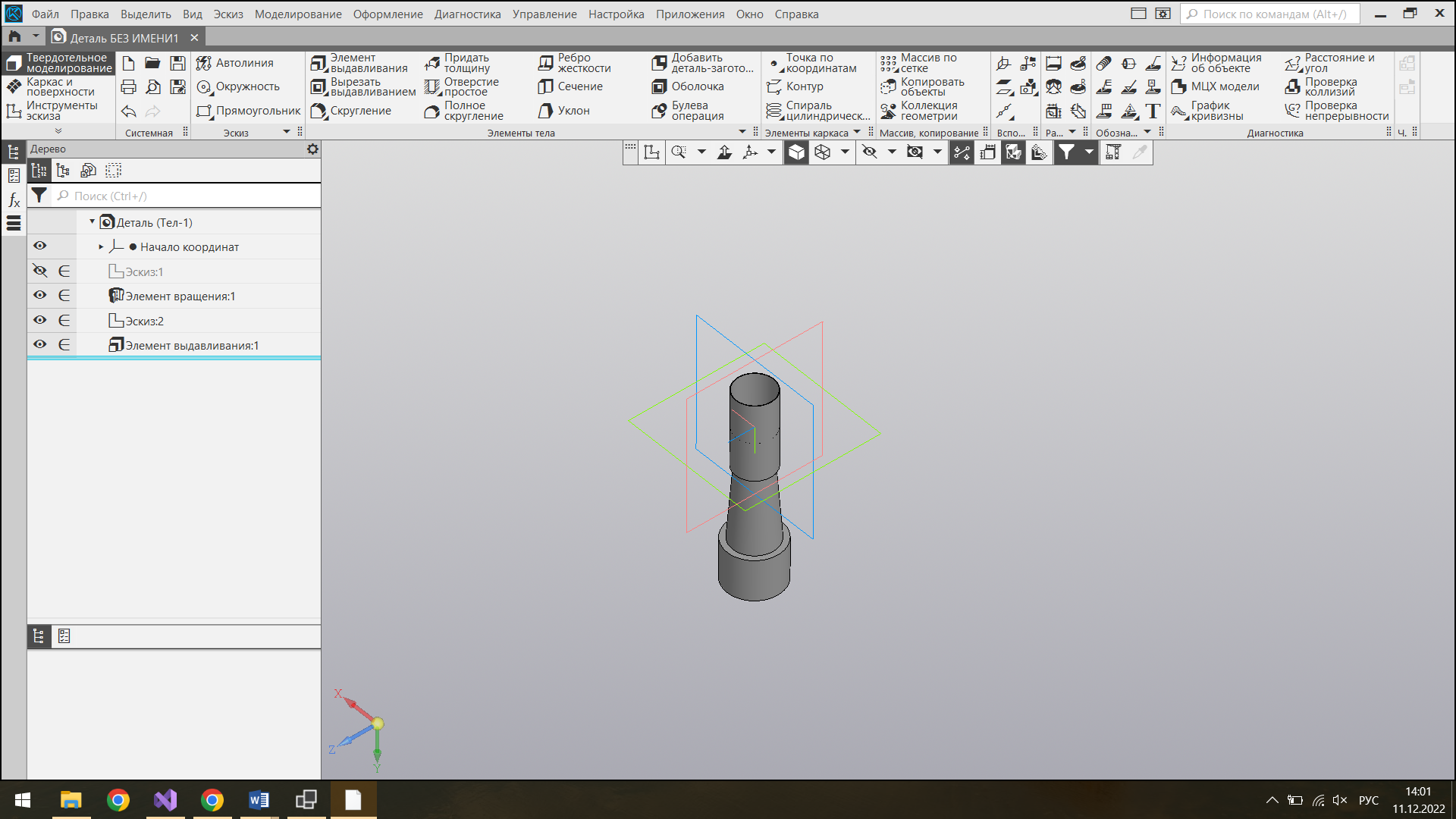


Рисунок 7.1 – Модель ладьи с минимальными введенными параметрами

На рисунке 7.2 представлена модель ладьи с максимальными введенными параметрами.

Максимальные параметры модели:

– Высота фигуры (А) – 600 мм;

– Диаметр нижнего основания (B) – 200 мм;

– Диаметр верхнего основания (C) – 100 мм;

– Высота нижнего основания (D) – 100 мм;

– Высота верхнего основания (E) – 90 мм.

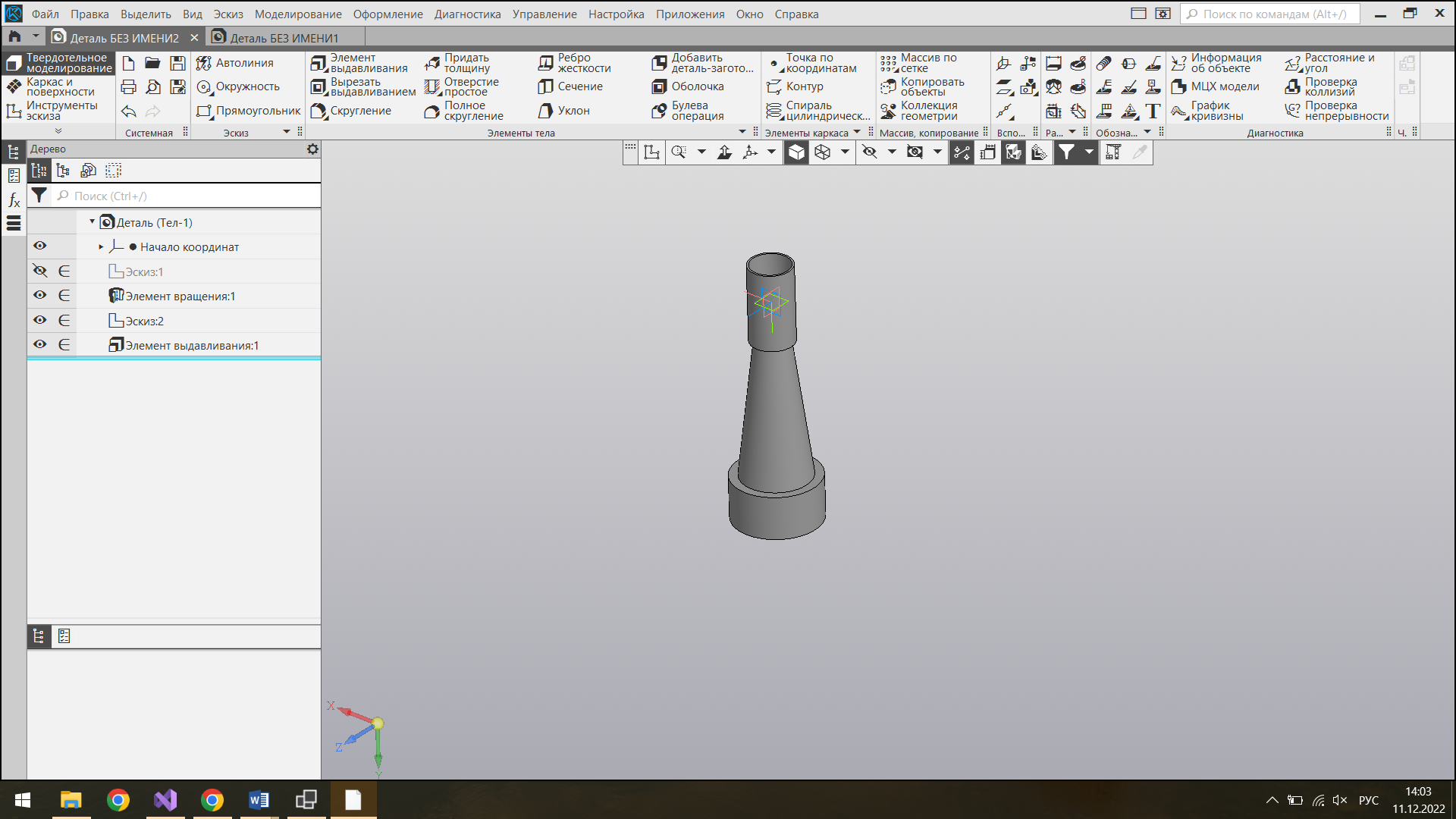


Рисунок 7.2 – Модель ладьи с максимальными параметрами

## 7.2 Модульное тестирование

В целях проверки корректности работы методов и свойств классов при помощи тестового фреймворка NUnit версии 3.13 проведено модульное тестирование, проверялись открытые поля, свойства и методы. Были протестированы классы модели: RookInfo, Point. На рисунке 7.3 представлена информация о модульном тестировании программы. Покрытие модели тестами составило сто процентов, что показано на рисунке 7.4.

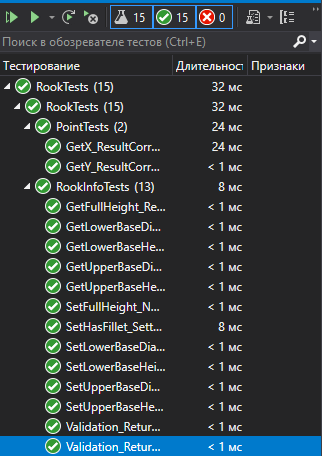


Рисунок 7.3 – Модульное тестирование плагина

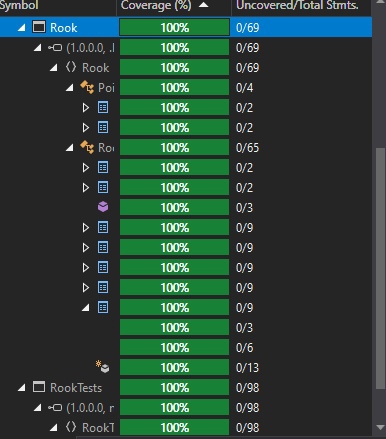


Рисунок 7.4 – Покрытие кода тестами

Описание тестов класса RookInfo и их параметров представлено в таблице 7.1.

Таблица 7.1 – Описание тестов RookInfo.

|  |  |
| --- | --- |
| **Название теста** | **Описание** |
| GetFullHeight\_ResultCorrect() | Тест геттера и сеттера свойства FullHeight |
| GetUpperBaseHeight\_ResultCorrect() | Тест геттера и сеттера свойства UpperBaseHeight |
| GetLowerBaseHeight\_ResultCorrect() | Тест геттера и сеттера свойства LowerBaseHeight |
| GetLowerBaseDiameter\_ResultCorrect() | Тест геттера и сеттера свойства LowerBaseDiameter |
| GetUpperBaseDiameter\_ResultCorrect() | Тест геттера и сеттера свойства UpperBaseDiameter |
| SetFullHeight\_NotSetted() | Тест сеттера свойства FullHeight |
| Validation\_ReturnsFalse() | Тест метода Validation. Он должен вернуть false |
| SetUpperBaseHeight\_NotSetted() | Тест сеттера свойства UpperBaseHeight |
| SetLowerBaseHeight\_NotSetted() | Тест сеттера свойства LowerBaseHeight |
| SetLowerBaseDiameter\_NotSetted() | Тест сеттера свойства LowerBaseDiameter |
| SetUpperBaseDiameter\_NotSetted() | Тест сеттера свойства UpperBaseDiameter |
| SetHasNewFeatures\_Setted() | Тест сеттера свойства HasNewFeatures |
| SetHasAnotherFeatures\_Setted() | Тест сеттера свойства HasAnotherFeatures |
| Validation\_ReturnsTrue() | Тест метода Validation. Он должен вернуть true |

Описание тестов класса Point и их параметров представлено в таблице 7.2.

Таблица 7.2 – Описание тестов класса Point.

|  |  |
| --- | --- |
| **Название теста** | **Описание** |
| GetX\_ResultCorrect() | Тест геттера и сеттера свойства X |
| GetY\_ResultCorrect() | Тест геттера и сеттера свойства Y |

## Нагрузочное тестирование

В целях проверки производительности работы плагина, было проведено нагрузочное тестирование. Тестирование производилось на ПК со следующей конфигурацией:

– ЦП Intel Core i5-1035G1 CPU, 2.5 ГГц;

– 8 Гб ОЗУ;

– Объем графической памяти 4 Гб.

Для нагрузочного тестирования был задан бесконечный цикл построения детали. Для измерения времени был использован класс Stopwatch. Тестирование заключалось в построении ладьи со стандартными параметрами. На рисунке 7.5 представлен график зависимости загруженности памяти от количества построенных деталей.

Стандартные параметры модели:

– Высота фигуры (А) – 70 мм;

– Диаметр нижнего основания (B) – 20 мм;

– Диаметр верхнего основания (C) – 15 мм;

– Высота нижнего основания (D) – 15 мм;

– Высота верхнего основания (E) – 14 мм.

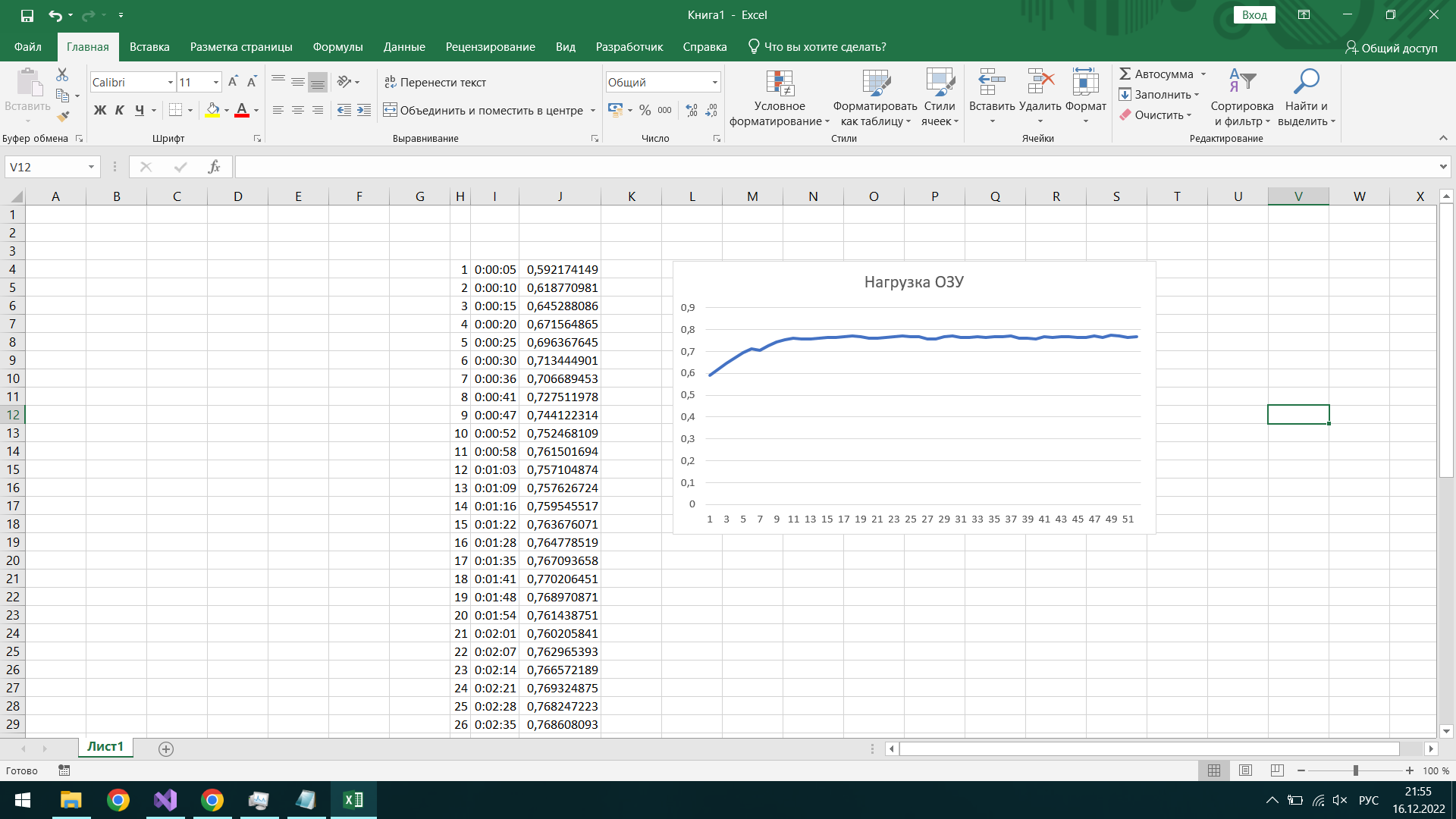


Рисунок 7. 5 – График зависимости затраченной памяти в Мб от количества моделей ладьи

На рисунке 7.6 представлен график зависимости времени построения от количества построенных деталей.

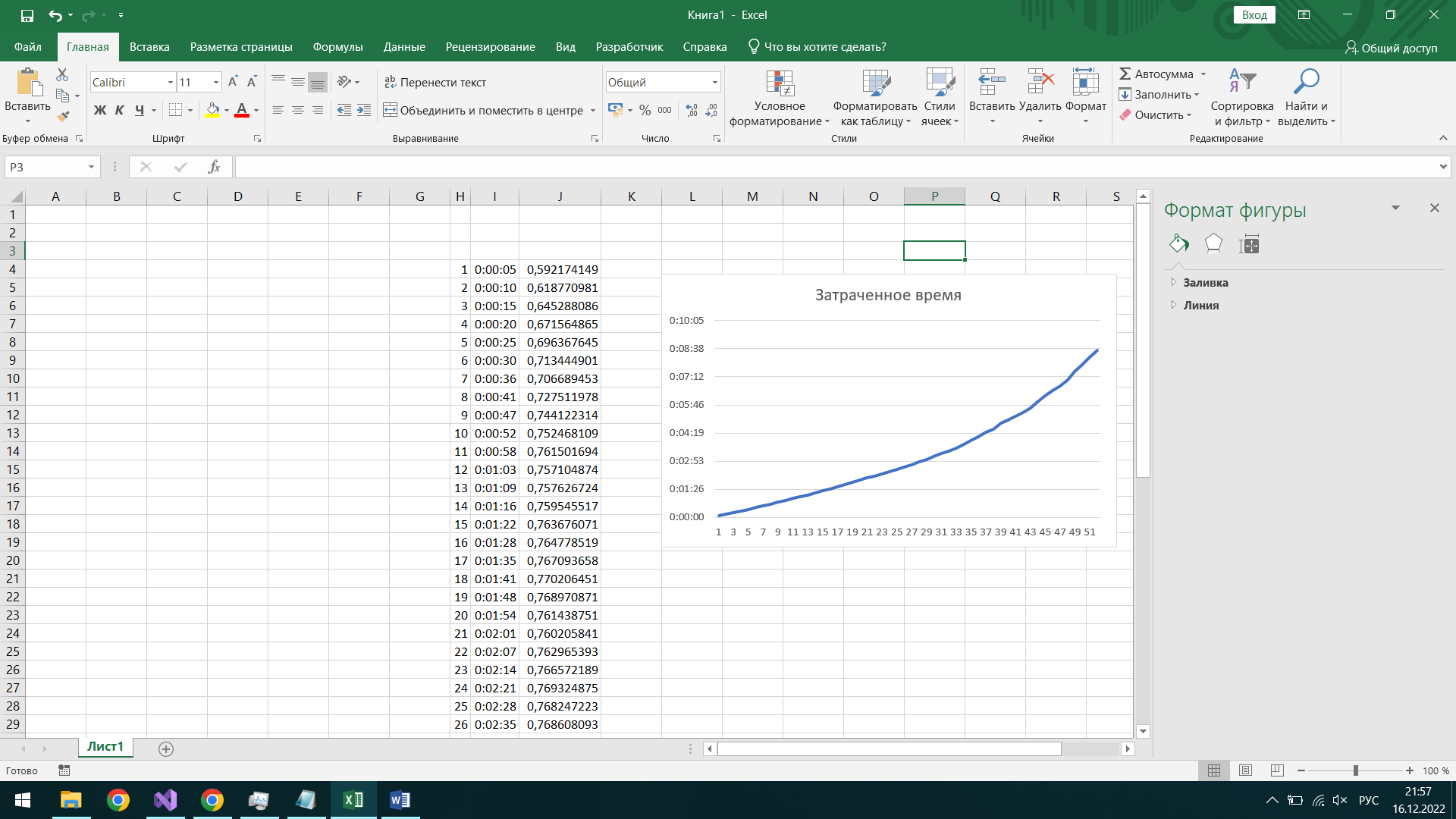


Рисунок 7. 6 – График зависимости затраченного времени от количества моделей ладьи

Тестирование длилось почти 9 минут, за которые было построено 51 модель ладьи.

По графику затраченного времени есть скачки в построении модели: так, в начале на ее построение застрачивалось 5-7 секунд, к концу время увеличилось до 15-19 секунд.

По графику затрачиваемой памяти видно, что примерно на 10 модели постепенно начинается освобождение небольших объемов данных для того, чтобы было возможно продолжать работу. Можно предположить, что в этот момент произошел переход в виртуальную память, то есть использование файла подкачки.

Виртуальная память — метод управления памятью компьютера, позволяющий выполнять программы, требующие больше оперативной памяти, чем имеется в компьютере, путём автоматического перемещения частей программы между основной памятью и вторичным хранилищем (например, твердотельным накопителем).

# Заключение

При выполнения лабораторных работ были изучены основные этапы проектирования программного продукта и его реализации, предметная область объекта проектирования, SDK «КОМПАС-3D». Было составлено техническое задание, разработан проект системы, составлены UML диаграммы классов, разработан макет пользовательского интерфейса.

В результате работы был разработан и реализован плагин для САПР «КОМПАС-3D», выполняющий построение 3D-модели ладьи по заданным параметрам.

Над реализованным плагином были проведены функциональное, модульное и нагрузочное тестирование на платформе Windows 10.

# Список использованных источников

1. Автоматизация вычислительных процедур в прикладных задачах инженерного проектирования [Электронный ресурс]. – URL: https://scienceforum.ru/2014/article/2014000201 (дата обращения: 20.10.2022).
2. Visual Studio [Электронный ресурс]. – URL: https://visualstudio.microsoft.com/ru/ (дата обращения: 20.10.2022).
3. Базовые интерфейсы API системы КОМПАС [Электронный ресурс]: PROGRAMMING-LANG.COM – режим доступа к статье: <http://programming-lang.com/ru/comp_soft/kidruk/1/j196.html>
4. КОМПАС-3D для разработчиков. [Электронный ресурс]. – URL: https://kompas.ru/solutions/developers/ (дата обращения: 20.10.2022).
5. Бесплатный аналог плагина [Электронный ресурс]: MECHANICAL.RU – режим доступа к статье: https://www.mechanical.ru/programs/mechanics/ (дата обращения: 20.10.2022)
6. NUnit [Электронный ресурс]. – URL: https://nunit.org/ (дата обращения: 20.10.2022).
7. Введение в Windows Forms [Электронный ресурс]. – URL: https://metanit.com/sharp/windowsforms/1.1.php (дата обращения: 20.10.2022).
8. Введение в UML от создателей языка. / Г. Буч, Д. Рамбо, И. Якобсон, 2-е изд. – ДМК Пресс, 2015 – 496 с.